2.2 VEHICLE MOVEMENT

Flight Processing (FP) process performs two basic functions: moving platforms and reporting updated state vectors (position, velocity and in some cases orientation and status data). These functions are executed for each simulation time interval.

For each time step, Flight Processing begins when engagement records are received from the C3I process. Each engagement record contains the platform identifiers and commands that slave the actions of the FP process to the battle management and engagement modeling functions of the C3I process. The platform identifiers are used to determine the acting and target platforms. The command portion of the record identifies the action to be taken.

The FP processes all the movement of ground platforms, and of missile, airplane, and helicopter flight. The actions identified through the C3I interface determine the current movement mode of each platform. For example, a vector command from the C3I process causes the FP process to vector an aircraft to the current position of the target platform.

The FP process reports the updated status of each platform to the other run-time processes over the sockets established with the FP process. The C3I process uses this information for the current position and velocity of the platforms. The Detection and Propagation processes use this information for the current state and status of platforms, for example, in determining dead and inactive platforms.

The functions performed by Flight Processing include aircraft flight, modes of movement, missile flight, ground platform movement, satellite flight, and interceptor missile flight.

Flight Modeling (Aerodynamic)

There are two aspects of flight modeling in FP. The first is determining where the aircraft is to fly and is handled through nine separate flight modes. Four flight modes are user defined: Waypoints, Wingman, Scramble, and Airborne Refueling. For Waypoints and Wingman modes, an aircraft will remain in these initial modes until the end of the scenario unless the platform's ruleset initiates another flight mode. Five modes are scheduled by rulesets: Engage, Drag, Return to Base (RTB), Avoid and Vector. Each of these flight modes are discussed in subsequent sections.

The second aspect of aircraft flight modeling is the set of algorithms used to update aircraft states and status. The equations describing aircraft flight are basically the same for all flight modes. These include aircraft and helicopter acceleration calculations, direction of flight propagation, altitude monitoring and terrain following and constant altitude rate flight.

Aircraft Flight

Both airplanes and helicopters are flown using a half-second update time on a modified Three Degrees Of Freedom (3-DOF) model of motion. The modeling accounts for the effects of gravity and atmosphere when calculating the acceleration force on the aircraft. The forces of drag, gravity, and thrust are summed to calculate the resulting acceleration acting on both types of aircraft.

Airplane Acceleration

Airplane drag is computed as a function of air density and lift. The air density is computed using pressure curve fit and temperature gradient data derived from the 1962 Standard Atmosphere data. Acceleration is determined by summing forces along the velocity vector using the calculated thrust and dividing by the initial aircraft mass, as follows:

$$A = (T - D - W^*\cos\varphi) / m$$

where

A	=	Actual acceleration (m/s^2)
T	=	Aircraft thrust (N)
D	=	Aircraft drag (N)
W	=	Aircraft weight (N)
φ	=	Angle between velocity vector and local vertical (rads)
m	=	Aircraft mass (kg)

After determining the aircraft's acceleration, the commanded maneuvers dictate the flight path. These commanded maneuvers result in the establishment of a vector in the direction of desired flight. Depending on the flight mode, this destination vector is either a vector relative to the location of another platform or a vector derived from a desired direction of flight. Two types of flight paths can be flown to achieve the desired direction: a straight line and a curve.

Helicopter Acceleration

Like airplanes, helicopters are flown using a simple 3-DOF flight model. The model accounts for drag and weight forces when computing the required thrust to allow for acceleration along the direction vector. Unlike airplanes, however, thrust is required to balance both weight and drag since helicopters have a limited lift generation capability at best. Also for helicopters, thrust is limited by both the maximum producible thrust by the rotor disk, and by the amount of power available to generate that thrust.

The acceleration in the desired direction is recomputed to account for the possible change in the horizontal component of thrust:

$$a = \frac{(T_H - D_{HP}\sin\theta - D_{Prof}\sin\theta - W\cos\theta - D_{VP}\cos\theta)}{m}$$

where

a	=	Desired acceleration
T_H	=	Thrust along direction vector
D_{HP}	=	Parasitical drag in ENU horizontal
D_{Prof}	=	Profile drag of rotor disk
W	=	Helicopter weight
D_{VP}	=	Parasitical drag in ENU vertical
m	=	Helicopter mass

The resultant acceleration magnitude is then applied along the direction vector when the helicopter state vectors are updated.

Both airplanes and helicopters are propagated by applying the computed acceleration along the direction of flight. Application of acceleration depends upon whether the airplane or helicopter is in straight flight (the velocity vector is currently aligned with its direction vector) or in curved flight.

Airplane Straight-Line Flight

In determining a flight path, each of the flight modes establishes this destination vector. A platform will use a straight-line flight path if the angle between the platform's current velocity vector and the destination vector is less than 8 degrees. For this case, the assumption is made that only slight modifications to the current flight path are required to maintain a course in the desired direction. The platform's roll is set to zero. The current magnitude of the velocity is reoriented to be along the given destination vector. The desired acceleration is also oriented along this vector. The position and velocity of the aircraft are then updated using the current position, the reoriented velocity, and the acceleration along the destination vector.

$$\vec{x}_n = \vec{x}o + \dot{\vec{x}}_o dt + 0.5\ddot{\vec{x}}dt^2$$

$$\dot{\vec{x}}_n = \dot{\vec{x}}o + \ddot{\vec{x}}dt$$

where $\vec{x}_o$ = current position

$\dot{\vec{x}}_o$ = reoriented velocity

$\ddot{\vec{x}}_o$ = acceleration along destination vector

Airplane Curved Flight

A platform will use a curved flight path if the angle between the platform's velocity vector and the destination vector is equal to or greater than 8 degrees. This flight pattern is also controlled by the given destination vector. Under nominal conditions (Aircraft speed is at its cornering speed) the turn radius is computed as:

$$R_T = (V^2) / G_T$$

where

R_T = Turn radius (m)

V = Aircraft velocity (m/s)

G_T = Turn acceleration = g-force x 9.80665 (m/s^2)

The angle through which the aircraft turns is determined by the following equation, where the internal FP time step is 0.5 seconds.

$$\theta_T = (V/R_T)\ dt$$

where

θ_T = Turn angle (rads)
V = Aircraft velocity (m/s)
R_T = Turn radius (m)
dt = Internal FP time step = 0.5 (s)

The roll angle is determined as the angle between the plane in which the turn is made and the local vertical. The resulting roll value is adjusted by a factor of the determined force in the turn over the specified maximum for the aircraft.

$$\text{Roll} = \text{Roll} \times (G/G\text{max})$$

where

Roll = Roll angle (rads)
G = Gravitational force in turn
Gmax = Aircraft's maximum gravitational force

Helicopter Straight Line Flight

Just as for airplanes, the helicopter flight modes establish the direction that the helicopter needs to move. Straight-line flight will be used if the angle between the helicopter's current velocity vector and the destination vector is less than 8 degrees. Again, the assumption is made that only slight modifications to the current flight path are required to maintain a course in the desired direction.

The acceleration vector is applied to the standard equations of motion for position and velocity to propagate the helicopter forward:

$$P_x = P_x + V_x D_t + \frac{1}{2} A_x D_t^2$$
$$P_y = P_y + V_y D_t + \frac{1}{2} A_y D_t^2$$
$$P_z = P_z + V_z D_t + \frac{1}{2} A_z D_t^2$$

and

$$V_x = V_x + A_x D_t$$
$$V_y = V_y + A_y D_t$$
$$V_z = V_z + A_z D_t$$

where

P_n = Helicopter ECI position in nth - direction (m)
V_n = Helicopter ECI velocity in nth - direction (m/s)
D_t = Integration time step (sec)
A_n = Helicopter ECI acceleration in nth - direction (m/s^2)

Roll is set to zero during straight line flight.

Helicopter Curved Flight

Helicopters whose direction vectors are more than eight degrees off of the current velocity vector are flown through a coordinated turn to achieve the new heading. Once the helicopter is within the eight degrees, then the straight line flight algorithm explained above is used.

The turns are performed by computing the helicopter position at each integration time step through the turn, as a function of the commanded G and the helicopter's current velocity:

$$P_x = P_x + \Delta P_{X_x} + \Delta P_{Y_x}$$
$$P_y = P_y + \Delta P_{X_y} + \Delta P_{Y_y}$$
$$P_z = P_z + \Delta P_{X_z} + \Delta P_{Y_z}$$

where

P_n	=	Helicopter position in ECI n - th direction (m)
ΔP_{Xn}	=	Local x positional change in ECI n - th direction (m)
ΔP_{Yn}	=	Local y positional change in ECI n - th direction (m)

Likewise, the new velocity is computed by applying the current velocity magnitude in the directions established by the local x and y velocity directions:

$$V_x = V_{Mag}\,\Delta V_{X_x} + V_{Mag}\,\Delta V_{Y_x}$$
$$V_y = V_{Mag}\,\Delta V_{X_y} + V_{Mag}\,\Delta V_{Y_y}$$
$$V_z = V_{Mag}\,\Delta V_{X_z} + V_{Mag}\,\Delta V_{Y_z}$$

where

V_n	=	Helicopter velocity in ECI n - th direction (m/s)
V_{Mag}	=	Velocity magnitude (m/s)
ΔV_{Xn}	=	Local x velocity in ECI n - th direction
ΔV_{Yn}	=	Local y velocity in ECI n - th direction (M)

The velocity magnitude used accounts for any acceleration or deceleration performed by the helicopter within the current integration step. DPxn, DPyn, DVxn, DVyn are the incremental position and velocity changes from the previous state and have been transformed from ECI coordinates to a local horizontal reference frame.

The roll of the helicopter is computed assuming the helicopter performs a coordinated turn. In a coordinated turn, thrust balances the weight of the helicopter in the vertical direction, and balances momentum forces in the horizontal direction. Thus, the force balancing equations are:

$$W = T\cos(\phi)$$

$$W\frac{V_{Mag}^{2}}{9.81R_t} = T\sin(\phi)$$

where

W	=	Helicopter weight (N)
T	=	Thrust (N)
ϕ	=	Roll angle (rads)
V_{Mag}	=	Velocity magnitude (m/s)
R_t	=	Turn radius (m)
9.81	=	Acceleration due to gravity (m/s^2)

Combining these two equations and solving for roll angle yields:

$$\phi = \tan^{-1}\left(\frac{V_{Mag}^{2}}{9.81R_t}\right)$$

where

ϕ	=	Roll angle (rads)
V_{Mag}	=	Velocity magnitude (m/s)
R_t	=	Turn radius (m)
9.81	=	Acceleration due to gravity (m/s^2)

This is the roll angle for the helicopter. If the helicopter is turning to the left, then the negative of this value is used.

Flight Modes

Waypoint Mode

The waypoints of a given aircraft provide the nominal flight path for the platform. The aircraft will follow these user-defined points unless interrupted by the C3I battle management functions. A waypoint consists of the altitude, speed, on time, off time, terrain following information, waypoint type, and the waypoint coordinate in either latitude and longitude or Military Grid Reference System (MGRS) units. Two alternative methods are available to the user for moving an aircraft between way points: aerodynamic and non-aerodynamic. The aerodynamic method uses thrust, gravity, lift and drag to calculate the aircraft accelerations required to achieve a desired speed. The non-aerodynamic integration uses simple aircraft position and speed to determine the aircraft state at each step.

Wingman Flight Mode

The wingman flight mode is specified by the user by designating a flight leader for an airborne platform. The wingman mode can be changed to other flight modes by the C3I process which returns wingmen to wingman mode after an engagement or a drag maneuver has been completed. The wingman mode is also used for airborne refueling operations. When a receiver and its flight are flying in formation with a tanker during the actual fuel transfer, they are flying as wingmen to the tanker. Even if the tanker is flying as a wingman to some tanker flight leader, it becomes a refueling flight leader to those receivers that are refueling.

Each wingman's position in the flight is defined by a position vector relative to the Flight leader. The wingman formation can be either the default formation, or a formation that is created by the user. Additionally, the refueling flight leader can have a special refueling formation defined for placement of those receiver aircraft that are refueling. If none exists, then the refueling aircraft placement will be based on the default formation as well.

Airborne Refueling Mode

There are two types of airborne refueling operations available: Mission Planned Refueling and Dynamic Refueling. Mission Planned Refueling allows preplanned refueling of platforms in a scenario prior to the completion of the other phases of their mission. Dynamic Refueling allows for refueling of platforms in a scenario when they are getting low on fuel.

Engage Flight Mode

The engage flight mode is selected by the C3I process for an aircraft to fly a path toward a selected target. There are three aircraft propagation methods for the engage flight mode: standard engage mode propagation, bomber propagation, and fpole propagation for aircraft that have launched a semi-active weapon against a target with air-to-air capability.

The standard engage propagation is used for flying all non-bomber aircraft that are vectored to engage ground or airborne targets, including TBMs. This method computes the attacker's velocity to approach the target, performs a predictive intercept for ABT threats, computes a direction of flight, chooses whether to fly a straight line or curved path to the target and updates the attacker's position and velocity after propagation is complete.

Bomber propagation is used if the attacker is a bomber engaging a ground target. The altitude will be adjusted according to the flight method used and the bomber will be flown for each internal time step of 0.5 seconds within the scenario interval. The bomber's altitude is adjusted to a commanded altitude if terrain following flight is selected. Otherwise, the altitude is the bomber's current altitude.

The FPole propagation is used to fly aircraft that have launched a semi-active missile at the target aircraft. It flies the attacker such that the target is kept within a user specified azimuth limit at all times to minimize closing rate while keeping the target in track. Elevation adjustments are applied when necessary to keep the target within a user specified elevation limit as well.

Scramble Flight Mode

The Scramble flight mode is employed when an aircraft has been scripted to leave its assigned home base or when the C3I process initiates this action in response to a command center's request to refill a Combat Air Patrol that is leaving its waypoint set. Scramble mode is used to transfer a flight of aircraft from its assigned airbase to an altitude of 70%

of the first waypoint altitude of the flight. Upon achieving that altitude the flight transitions to standard waypoint mode.

Drag Flight Mode

The drag flight mode is used to compute an escape direction vector for an aircraft that is being engaged by an attacking aircraft. It allows the user to choose whether the fleeing aircraft should perform a drag or a beam maneuver, and what kind of beam maneuver to perform. It also ensures that the fleeing aircraft flies within an altitude corridor specified by the user.

Return to Base (RTB) Flight Mode

The RTB flight mode is used for aircraft that have run out of weapons, targets, fuel or time. The RTB Flight mode flies aircraft back to their home airbase, i.e., their first waypoint. The RTB mode expects aircraft to start from a normal flight mode, such as waypoints, rather than a reaction flight mode, such as Engage or Drag. When a platform is scheduled for the RTB mode, it is explicitly put into a normal flight mode.

Avoid Flight Mode

The Avoid mode of flight for aircraft causes the aircraft to maintain a flight path that is normal to the vector to the aircraft being avoided. This flight mode is currently used only by the Wild Weasel ruleset. The Wild Weasel will optionally fly this maneuver during the lock phase of an engagement. The Wild Weasel also makes use of this flight mode in a reaction to being locked by a SAM system.

Vector Flight Mode

This flight mode flies a fighter in a direction rather than in a pursuit course after another platform. The C3I process computes an intercept point (IP) for an incoming airborne platform, and vectors the fighter by transferring (over the socket) the heading, the ground range to the interception point, the altitude, and the speed the fighter should fly to the FP process. The decision to use the Vector flight mode is made by C3I based on the fighter’s ruleset. This flight mode is only used by the Fighter ruleset.

Processing of Captive Platforms

Captive platforms are created by placing a specific platform on the target list of either a Bomber or Fighter-Bomber host platform and specifying a waypoint at which the captive platform will be launched from the host platform. A captive platform is a pseudo-flight mode in which the RCS of the captive is added to that of the host. In captive mode the guest platform is inactive.

Low-Level Transit Routes (LLTRs)

EADSIM models Low-Level Transit Routes (LLTRs) across a Missile Engagement Zone (MEZ). These routes allow for the safe passage of aircraft within a MEZ.

Aircraft will examine their flight routes at transitional times to see if the planned flight route will carry the aircraft within a MEZ that the aircraft knows about. Aircraft which determine that their planned flight route will carry them within a MEZ will search for and adopt an LLTR which allows them safe passage. If no LLTR is found which is suitable then the aircraft will proceed through the MEZ.

Once an LLTR has been adopted, the LLTR is simply followed as a set of waypoints. Normally, an aircraft will not depart the transit route once it is on the route. Aircraft may also adopt and then abandon an LLTR prior to reaching it.

Profile Deployment

Deployment of profile waypoint sets provides a dynamic capability for adopting a certain flight profile when a platform engages against a ground target. It allows the platform to leave its original waypoint set and follow the profile waypoints until within range of the target to engage. Once the engagement has been completed, the platform can then return to its original set of waypoints. Since the profile waypoints are set up at the ruleset level, the same flight profile can be used by multiple aircraft within the scenario regardless of which target is being engaged and where that target is located within the scenario.

Missile Flight

The ballistic missile model provides the capability to represent multi-stage missiles with detailed pitch program guidance. Flight section options include the ability to set up multiple powered flight segments representing engine thrusting, unpowered segments representing ballistic flight, and missile staging events representing missile mass changes. All of these options can be timed or delayed to realistically model actual missile weapon systems. Guidance options such as minimum energy, depressed and lofted flyouts, gravity turns, or multiple guidance phases can be used to achieve the desired flyout.

The ballistic missile model is based on the Powered Flight Program (PFP). The PFP provides the missile flight modeling for the ATTACK program which is used to create community-standard threat tapes. The PFP version used to create the ballistic missile model in EADSIM is described in Technical Report, CS85-SPACECMD-50, Powered Flight Program (PFP), dated April 1985, prepared by Teledyne Brown Engineering, Contract No. F05604-82-C-0061. The original FORTRAN methodology was restructured to update multiple missiles to the same time and provide the missile states to the other models. The new code was further analyzed and significant speed improvements were achieved over the original PFP. These improvements primarily came through more efficient implementation of equations.

The ballistic missile model is further supported by the launch iteration schemes. The launch iteration schemes determine the correct setting of specific parameters to allow the missile to fly the desired range to the target. The launch iteration schemes provide the capability to model the guidance options mentioned above. These algorithms were developed specifically for application in the EADSIM.

Ground Platform Movement

Ground platforms move between waypoints at speeds specified for each waypoint. Each platform waypoint includes an “on” time and an “off” time; the on time determining when the platform becomes active at the waypoint and the off time determining when the platform will leave the waypoint. A platform can be scripted to move or it can be commanded to move. The same waypoint information is required for commanded movement as for scripted movement but a commanded platform must have at least two waypoints specified (This is because the commanded ground platform, always an SSFU, must move to its launch site from its initial position).

Ground platform movement is along the line defined by successive waypoints at the speed specified by the destination waypoint. The platform's position is updated each update interval until it reaches the destination waypoint. Each ground waypoint position is maintained internally in ECI coordinates and the updated position is calculated using the formula,

> Position vector = Unit vector of updated position vector * (Earth Radius + Elevation for map coordinates)

Where the Elevation for map coordinates is taken at the destination waypoint. The platform will remain at the destination waypoint until the waypoint off time expires. At that time it will proceed to the next waypoint.

Satellite Flight Modeling

A satellite's motion in EADSIM is modeled by Keplerian equations of motion for orbital bodies. These equations have been implemented in a program known as Tarpre2 (Target Predictor, version 2) as described in Methods of Orbit Determination by Escobal. Tarpre2 simulates an elliptical satellite flight path and uses an initial satellite state vector and a time delta to compute each new state vector of the satellite along its path. An object can be propagated to any part of its elliptical path in a single step. The delta time may be entered as positive or negative, allowing the satellite to be propagated forward or backward. The time delta is incremented by the length of the simulation integration step at each simulation update time. By using the original state vector and a new increment to update the satellite state for each time step, the errors caused by the propagation techniques are not compounded.

Interceptor Missile Flight

Two propagation methods are used to model explicit SAM interceptor missile flyout in EADSIM: Constant Velocity Non-Aerodynamic Flight and 3 Degree of Freedom Aerodynamic Missile Flight. Constant Velocity Non-Aerodynamic Flight a constant velocity or a flyout table specified for the weapon by the user to propagate the missile in 3 dimensions. The state vector for the missile is updated at the end of each integration interval. Positional changes in the missiles state vector are computed in straight line segments at a constant velocity. If the Constant Velocity option is used, then the velocity contained in the weapon record is used to propagate the missile forward. If a flyout table is used, the table is accessed at the time of missile launch as a function of the target's ground range and altitude with respect to the Flexible SAM launching the weapon. The resultant time of flight from this table is converted into a velocity based on the range to the target. This velocity is used to propagate the missile forward. Turns are instantaneous.

The model supports command guided, non-homing flight as well as semi-active/active homing flight A command guided missile flies to a point in space specified by C3I. A semi-active/active missile homes on the target position (i.e., guidance is a function of track position and velocity).

The 3DOF aerodynamic missile model provides a realistic representation of the trajectories associated with boost, flyout, and homing stages of an interceptor engaging a target. Guidance options during flyout include ballistic, following a prescribed flight path angle profile, or guiding to a point in space for both powered and unpowered flight. Proportional navigation is used to steer the missile toward the target during endgame.

The flight trajectories during flyout and homing are realistically constrained with aerodynamic, structural, and propulsion capabilities. Lateral acceleration for steering is aerodynamically limited according to a user-specified maximum angle-of-attack. Also a "hard" limit is placed on lateral steering acceleration representing maximum g-loading structural capabilities. During homing, a special exoatmospheric steering option is included for modeling lateral (or divert motor) thrusters that act through the homing section's center-of-gravity. If this option is selected, the lateral thrusters automatically augment aerodynamic steering if angle-of-attack limits are reached. The magnitude and duration of thrust from the divert motors are constrained by a user-specified acceleration limit and divert velocity.

2.2.1 Functional Element Design Requirements

Not currently available.

2.2.2 Functional Element Design Approach

This section describes the design elements that implement the vehicle movement design requirements.

Design Element 2-1: Airplane Straight-Line Flight

In determining a flight path, each of the flight modes establishes its destination vector. A platform will use a straight-line flight path if the angle between the platform's current velocity vector and the destination vector is less than 8 degrees. For this case, the assumption is made that only slight modifications to the current flight path are required to maintain a course in the desired direction. The platform's roll is set to zero. The current magnitude of the velocity vector is reoriented along the given destination vector. The desired acceleration is also oriented along this vector. The position and velocity of the aircraft are then updated using the current position, the reoriented velocity, and the acceleration along the destination vector.

$$\vec{x}_n = \vec{x}_o + \vec{x}_o\,dt + 0.5\,\vec{x}\,dt^2$$

$$\vec{x}_n = \vec{x}_o + \vec{x}\,dt$$

where

$\vec{x}_o$ = current position

$\dot{\vec{x}}_o$ = reoriented velocity

$\ddot{\vec{x}}_o$ = acceleration along destination vector

If the velocity of the aircraft is less than the user-specified minimum speed for the aircraft, the destination vector is modified before the state is updated. If the aircraft still has fuel, the aircraft is assumed to still have thrust. The destination vector is set equal to the difference between the unit vector of the destination vector and the unit vector of the position. This causes the aircraft to descend as it gathers speed. For the case where the aircraft has no fuel, the destination vector is oriented straight down. The modified destination vector is then used as the direction of the velocity and the acceleration to compute the new position and velocity of the aircraft.

Design Element 2-2: Airplane Curved Flight

A platform will use a curved flight path if the angle between the platform's velocity vector and the destination vector is equal to or greater than 8 degrees. This flight pattern is also controlled by the given destination vector.

The lower limits of the velocity are tested to verify that the aircraft is not below it's minimum speed. If the velocity is below minimum speed, the destination vector is modified to point straight down. The acceleration of gravity is applied in the direction of the destination vector, ignoring the effects of drag. The position and velocity are updated using the current velocity vector and the acceleration of gravity as:

$$\vec{x}_n = \vec{x}_o + \vec{x}_o\, dt + 0.5\, \vec{x}\, dt^2$$

$$\vec{x}_n = \vec{x}_o + \vec{x}\, dt$$

where

$\vec{x}_o$ = current position

$\dot{\vec{x}}_o$ = reoriented velocity

$\ddot{\vec{x}}_o$ = acceleration along destination vector

Roll is assumed to be zero for this case.

If the aircraft is flying faster than it's minimum speed, the required radius for the turn is computed based on the aircraft's current velocity and the gravitational force specified for the turn. If the current velocity is less than the aircraft's cornering speed, the pre-computed turn parameters and the current velocity are used to determine the radius of the turn; otherwise, the radius of the turn is determined by the aircraft's maximum gravitation force. If the current velocity is less than the corner speed, the aircraft's turn rate is computed from a quadratic curve fit to velocity. The radius of the turn is then computed to be the magnitude of the velocity divided by the turn rate. If the turn rate is negative, the aircraft's turn rate is set to .01 and the turn radius will be set to the velocity divided by 0.01. The user is warned of the bad turn parameters in the log file for flight processing. If the aircraft's speed is currently greater than the corner speed, the turn radius is computed as:

$$R_T = (V^2)/G_T$$

where: R_T = Turn radius (m)
V = Aircraft velocity (m/s)
G_T = Turn acceleration = g - force x 9.80665 (m/s^2)

The angle through which the aircraft turns is determined by the following equation, where the internal FP time step is 0.5 seconds.

$$\theta_T = (V/R_T)dt$$

where θ_T = Turn angle (rads)
V = Aircraft velocity (m/s)
R_T = Turn radium (m)
dt = Internal FP time step = 0.5 (s)

The roll angle is determined as the angle between the plane in which the turn is made and the local vertical. The resulting roll value is adjusted by a factor of the determined force in the turn over the specified maximum for the aircraft.

$$Roll = Roll(G/G_{max})$$

where Roll = Roll angle (rads)
G = Gravitational force in turn
G_{max} = Aircraft/s maximum gravitational force

Design Element 2-3: Vector

Platform movement for the vector flight mode is performed in 0.5 second integration steps, up to the end of the update interval. At the beginning of the update interval, command speed V_{Cmd} is initialized to the desired speed sent across the socket from the C3I process. The magnitude of the fighter position vector is calculated so the fighter current altitude can be found:

$$H = P_{Mag} - R_E$$

where

H = Altitude of fighter (m)
P_{Mag} = Magnitude of fighter position vector (m)
R_E = Earth radius (m)

The desired direction of flight vector is calculated. First, the cross product of the fighter position vector and the intercept point position vector is found:

$$\vec{C} = \vec{P} \times \overrightarrow{IP}$$

where $\vec{C}$ = Cross product of $\vec{P}$ and $\overrightarrow{IP}$ (m)
$\vec{P}$ = Fighter position vector (m)
$\overrightarrow{IP}$ = Intercept point position vector (m)

Then, the resultant vector is crossed with the fighter position vector:

$$\overrightarrow{Dir} = \vec{C} \times \vec{P}$$

where $\overrightarrow{Dir}$ = New direction of vector (m)
$\vec{C}$ = Cross product of $\vec{P}$ and $\overrightarrow{IP}$ (m)
$\vec{P}$ = Fighter position vector (m)

The resultant vector is normal to the fighter position vector. It also points in the direction of the intercept point. This is the desired direction vector. If terrain following or an altitude adjustment is required, then the terrain following methodology is used to modify the direction vector. The floor altitude used by the terrain following is set by the commanded altitude. Due to the nature of the Vector flight mode maneuver, the altitude maintained will be measured MSL rather than AGL.

If terrain following or altitude adjustment is not necessary, then the direction vector is modified to account for Earth curvature by maintaining a constant altitude rate while flying the aircraft. If the direction vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, then it is adjusted by altitude monitoring. The ceiling altitude for altitude monitoring is defaulted to 100000 meters. The floor altitude is set to 200 meters.

Next, the fighter climbs or descends to the commanded intercept altitude while turning and proceeding toward the target. The climb/descent angle is limited in order to give a reasonable climb/descent. Once the fighter reaches the commanded altitude, it flies to the intercept point at that altitude.

Another property of the vector flight mode is that the fighter will fly to its intercept point, and then fly beyond the point to a re-attack range specified in the Vector Phase of the fighter ruleset. This allows the fighter to search for the attacker target for some distance after passing the intercept point. Consequently, the fighter must still fly along the current direction of flight after passing the intercept point (IP) until the re-attack range has been exceeded. Once the fighter passes the IP and the computed direction vector is in the opposite direction of the fighter direction of flight, the negative of the direction vector is used to guide the fighter.

If an aircraft located within a MEZ is being vectored at a heading to intercept a target aircraft, then it will follow the associated LLTR, if any, to exit the MEZ before flying to the IP point. Once out of the MEZ, it will transition to the Vector flight mode and fly to the IP as described above.

Design Element 2-4: Drag/Beam

The drag flight mode is used to compute an escape direction vector for an aircraft that is being engaged by an attacking aircraft. It allows the user to choose whether the fleeing aircraft should perform a drag or a beam maneuver, and what kind of beam maneuver to perform. It also ensures that the fleeing aircraft flies within an altitude corridor specified by the user.

The C3I process uses the drag flight mode if an aircraft is reacting to an engagement by a hostile aircraft. The C3I process breaks off the aircraft's current engagement and sends a drag command to FP. The FP process then determines whether the aircraft will fly a drag maneuver or a beam maneuver. Two conditions must be met for a drag maneuver to be executed: the angle threshold and the time-to-attacker threshold.

First, the line-of-sight vector from aircraft to attacker is computed according to:

$$\vec{LOS} = \vec{P}_{Atck} - \vec{P}_{AC}$$

where

$\vec{LOS}$ = Line-of-sight to attacker vector (m

$\vec{P}_{Atck}$ = Attacker position vector (m)

$\vec{P}_{AC}$ = Aircraft position vector (m)

Next, the cosine of the angle between the line-of-sight vector and the aircraft's velocity vector is computed by:

$$Cos_q = \vec{U}_{LOS} \cdot \vec{U}_{Vel}$$

where

Cos_q = Cosine of angle between line-of-sight and a/c velocity vecto

$\vec{U}_{LOS}$ = Unit line-of-sight vector to attacker

$\vec{U}_{Vel}$ = Attacker unit velocity vector

If the angle between the aircraft velocity vector and the line-of-sight vector is greater than the drag phase angle threshold, then the cosine of the angle is less than the cosine of the drag phase angle threshold and the first condition is met.

The range the aircraft can fly during the drag phase time threshold is computed by:

$$R_{Thresh} = V_{Mag} T_{Thresh}$$

where R_{Thresh} = Range aircraft flies in drag phase threshold time (m)

V_{Mag} = Magnitude of aircraft velocity (m/s)

T_{Thresh} = Drag phase threshold time (s)

If the time to reach the attacking aircraft is greater than the drag phase time-to-attacker threshold, then the magnitude of the line-of-sight vector is greater than the range the aircraft can fly during the time threshold and the second condition is met.

If both of these conditions are met, then the aircraft performs a drag maneuver. This is accomplished by the calculation of the drag maneuver direction vector according to:

$$\vec{D}_{Drag} = \vec{P}_{AC} - \vec{P}_{Atck}$$

where

$\vec{D}_{Drag}$ = Drag maneuver direction vector(m)

$\vec{P}_{AC}$ = Aircraft position vector (m)

$\vec{P}_{Atck}$ = Attacker position vector (m)

If one of the drag threshold conditions is not met, then the fleeing aircraft executes a beam maneuver. First, a vector pointing to the right of the aircraft position and perpendicular to both the line-of-sight vector and the aircraft position vector is computed:

$$\overrightarrow{D_{Beam}} = \overrightarrow{LOS} \times \overrightarrow{P_{AC}}$$

where $\overrightarrow{D_{Beam}}$ = Beam maneuver direction vector if attacker on left (m)

$\overrightarrow{LOS}$ = Line-of-sight vector (m)

$\overrightarrow{P_{AC}}$ = Aircraft position vector (m)

If the attacking aircraft is to the left of the fleeing aircraft, then this vector becomes the beam direction vector. Otherwise, the beam direction vector is recomputed in the opposite direction according to:

$$\vec{D}_{Beam} = -\vec{D}_{Beam}$$

where

$\vec{D}_{Beam}$ = Beam maneuver direction vector if attacker on right(m)

The user can adjust the beam direction vector by setting the drag phase beam angle to some angle other than 0.0. This angle is measured down from the horizontal plane defined by the beam direction vector. A vector in the direction 45 degrees down from the horizontal is first computed by:

$$\vec{D}_{45} = \vec{U}_{D_{Beam}} - \vec{U}_{P_{AC}}$$

where

$\vec{D}_{45}$ = Vector 45° down from horizontal plan

$\vec{U}_{D_{Beam}}$ = Unit beam maneuver direction vector

$\vec{U}_{P_{AC}}$ = Unit aircraft position vector

Next, this new vector is adjusted based on the user input beam angle according to the following conditions:

if $315° \le \theta_{Beam} \le 360°$: $D_{Beam_Z} = D_{45_Z}((\theta_{Beam} - 360)/45)$

or if $\theta_{Beam} \ge 225°$: $D_{Beam_X} = D_{45_X}((\theta_{Beam} - 270)/45)$

$D_{Beam_Z} = -D_{45_Z}$

or if $\theta_{Beam} \ge 135°$: $D_{Beam_X} = -D_{45_X}$

$D_{Beam_Z} = D_{45_Z}((180 - \theta_{Beam})/45)$

or if $\theta_{Beam} \ge 45°$: $D_{Beam_X} = D_{45_X}((90 - \theta_{Beam})/45)$

otherwise: $D_{Beam_Z} = D_{45_Z}(\theta_{Beam}/45)$

where D_{Beam_n} = Component of beam maneuver vector in nth-direction
D_{45_n} = Component of D_{45} vector in nth-direction
θ_{Beam} = Drag phase beam angle (deg)

After the beam angle adjustment, this vector becomes the beam direction vector that is used to propagate the aircraft movement during integration.

Prior to integration, the direction vector is set to either the drag maneuver direction vector or the beam direction vector. Next, if the direction vector causes the aircraft to fly above the ceiling altitude or below the floor altitude, then it is adjusted for altitude monitoring. The ceiling and floor altitudes for altitude monitoring is set to the user inputs specified in the Drag phase of the aircraft ruleset. The default values for the ceiling and floor altitudes are 100000 and 200 meters, respectively.

Once the direction vector is finalized, it is used to fly the aircraft for each internal time step of 0.5 seconds within the scenario interval. If the angle between the aircraft's velocity vector and the direction vector is less than 8 degrees, the aircraft will fly straight at maximum velocity along the new direction vector. Otherwise, the aircraft will fly in a curved path at the aircraft's maximum cornering speed in a direction toward the new direction vector.

Design Element 2-5: F-Pole

F-Pole propagation is used to fly aircraft that have launched a semi-active missile at a target aircraft. It flies the attacker such that the target is kept within a user-specified azimuth limit at all times to minimize closing rate while keeping the target in track. Elevation adjustments are applied when necessary to keep the target at the user-specified elevation limit as well.

After the target position has been updated to the current simulation time of the attacker, the command speed of the attacker is set. If its current speed is greater than the corner speed, command speed is set to the current speed. Otherwise, the command speed is set to the corner speed. Next, the azimuth and elevation limits are applied to the direction vector. To do this, the vector perpendicular to both the attacker position and the target relative position vectors is first computed:

$$\overrightarrow{P_x} = \overrightarrow{Rel} \times \vec{P}$$

where $\overrightarrow{P_x}$ = Local horizontal plane x-axis (m)
$\overrightarrow{Rel}$ = Target relative position vector (m)
$\vec{P}$ = Attacker position vector (m)

Next, the vector coplanar to the target relative position vector but perpendicular to the new x-axis vector is computed:

$$\overrightarrow{P_x} = \overrightarrow{Rel} \times \vec{P}$$

where $\overrightarrow{P_x}$ = Local horizontal plane x-axis (m)
$\overrightarrow{Rel}$ = Target relative position vector (m)
$\vec{P}$ = Attacker position vector (m)

The target z component relative to this local coordinate frame is then computed by:

$$Tgt_Z = \vec{U}_P \cdot \vec{U}_{Rel}$$

where
Tgt_Z = Local z-axis component of target position
$\vec{U}_{Rel}$ = Unit target relative position vector
$\vec{U}_P$ = Unit attacker position vector (local z-axis

The target elevation angle is then computed:

$$\phi_{Tgt} = \frac{\pi}{2} - \cos^{-1}(Tgt_z)$$

where ϕ_{Tgt} = Target elevation (rad)
Tgt_Z = Local z-axis component of target position

If the target elevation does not exceed the elevation limit, then a local direction vector is computed that points in the direction corresponding to the azimuth limit from the target relative position vector:

$$D_X = \sin(\theta_{Lim})$$
$$D_Y = \cos(\theta_{Lim})$$
$$D_Z = 0.0$$

where D_n = Direction vector component in local nth direction
θ_{Lim} = Azimuth limit angle (rad)

If the target elevation does exceed the elevation limit, then the local direction vector is computed so the azimuth and elevation of the target are at the azimuth and elevation limits from the target relative position vector:

$$D_X = \cos(\phi_{Lim})\sin(\theta_{Lim})$$
$$D_Y = \cos(\phi_{Lim})\cos(\theta_{Lim})$$
$$D_Z = \pm\sin(\phi_{Lim}) \text{ for } \pm\phi_{Tgt}$$

where D_n = Direction vector component in local nth direction
ϕ_{Lim} = Elevation limit angle (rad)
θ_{Lim} = Azimuth limit angle (rad)
ϕ_{Tgt} = Target elevation angle (rad)

This direction vector is computed in a different local frame than the one computed above. The x-axis is the same as the local x-axis above, but the y-axis is the target relative position vector and the z-axis is perpendicular to the x-axis and the target relative position vector. The z-axis is computed by:

$$\overrightarrow{P_Z} = \overrightarrow{P_X} \times \overrightarrow{Rel}$$

where $\overrightarrow{P_Z}$ = Local horizontal plane z-axis (m)
$\overrightarrow{P_X}$ = Local horizontal plane x-axis (m)
$\overrightarrow{Rel}$ = Target relative position vector (m)

The resultant direction vector from the azimuth and elevation adjustments is then rotated into ECI coordinates.

Since the azimuth limit can be applied in both positive and negative directions around the target relative position vector, a second ECI direction vector is computed using the local direction vector of:

$$D_{X_2} = -D_X$$
$$D_{Y_2} = D_Y$$
$$D_{Z_2} = D_Z$$

where
D_{n_2} = Direction vector component in local n-th 2nd directio
D_n = Direction vector component in local n-th direction

Commanded speed is then applied to a unit vector in both directions to compute the perspective attacker velocity. Then, closing rate is computed as the relative velocity difference between the attacker velocities in both directions and the target velocity. The direction vector that results in the smaller closing velocity is used to fly the aircraft.

When azimuth and elevation limit adjustments have been completed, the attacker direction vector is adjusted for the last time to account for altitude limits. The ceiling altitude for altitude monitoring is defaulted to 100000 meters. The floor altitude is set to 200 meters.

Once all direction vector adjustments are completed, the angle between the final direction vector and the aircraft current velocity vector is computed. If the angle is less than 8 degrees, the attacker will be flown straight toward the target. Otherwise, the attacker will be flown in a curve toward the target direction vector. This process is repeated in 0.5 second steps until the end of the update interval.

Design Element 2-6: Avoid

The Avoid mode of flight for aircraft causes the aircraft to maintain a flight path that is normal to the vector to the aircraft being avoided. When operating in the Avoid flight mode, the aircraft attempts to maintain a course normal to the platform being avoided i.e., the aircraft remains at a constant distance from the avoided platform. Within this flight mode, the aircraft may fly terrain following at the altitude commanded by the ruleset. The floor altitude used by the terrain following is set by the commanded AGL altitude.

If terrain following is not necessary, then if the direction of flight causes the aircraft to fly above the ceiling altitude or below the floor altitude, it is adjusted for altitude monitoring. The ceiling altitude for altitude monitoring is defaulted to 100000 meters. The floor altitude is set to 200 meters.

The target vector from the avoiding platform to the avoided platform is first computed. The avoidance vector that is normal to the target vector is next computed as the cross product of the target vector with the avoiding platform's position. This avoidance vector is the vector which defines the direction of flight for the platform. This avoidance vector will cause the platform to fly to the right while avoiding the platform. If the platform to be avoided is to the right of the avoiding platform's current velocity vector, the avoidance vector undergoes a 180-degree phase shift to cause the platform to fly to the left to avoid the platform.

2.2.3 Functional Element Software Design

Call Tree

The calling tree for vehicle movement in EADSIM is shown in Figure 2.2-1. UpdateOpfacs is called from the Flight Processing main program on each iteration of its processing loop. UpdateOpFacs invokes the appropriate integration model to move the platform over the next simulation interval. Flight is called to update aircraft state (i.e., position and velocity) vectors. The defensive and reactive maneuvers of interest for purposes of this document, avoid, vector, drag, beam and vector, are shaded in the figure. The drag and beam maneuvers are related and are both modeled in the FlyDrag procedure.

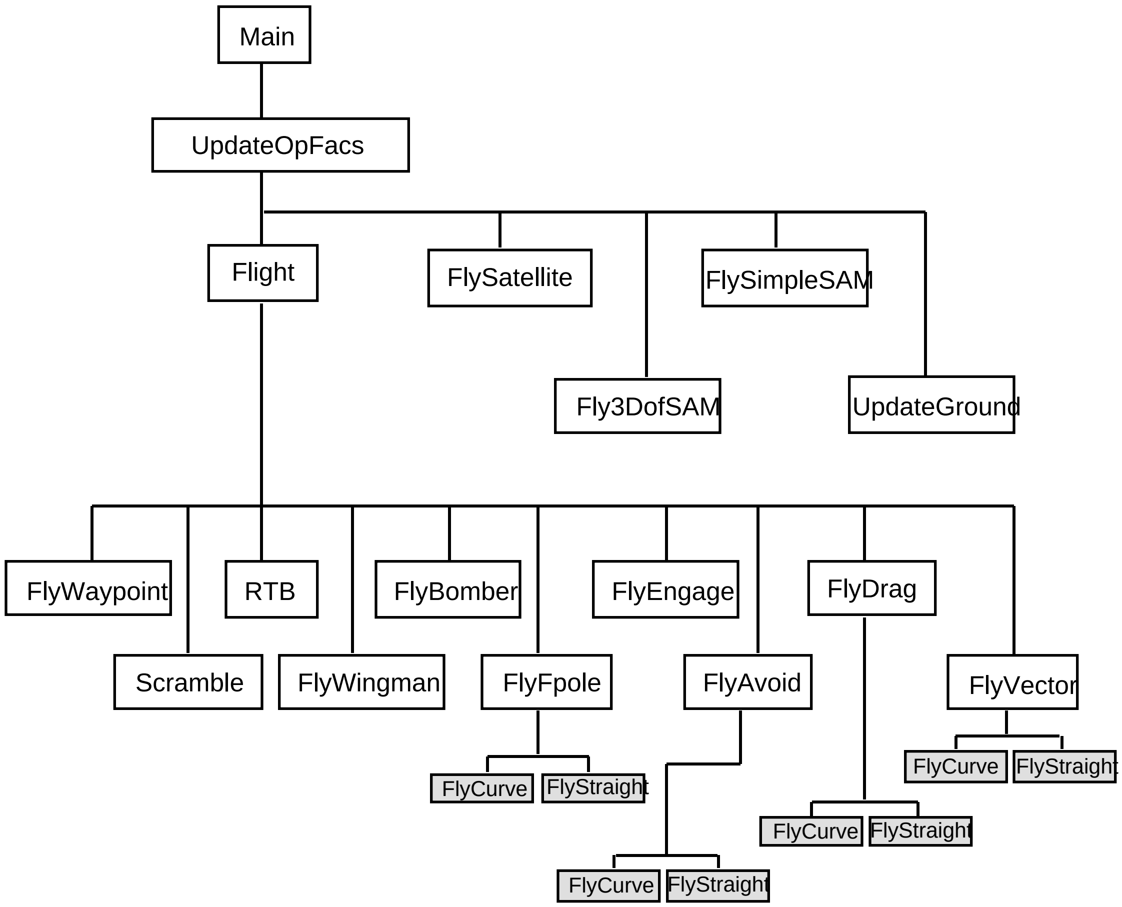

FIGURE 2.2-1. Calling Tree for Vehicle Movement.

Module Descriptions and Logic Diagrams

FlyStraight:

The Flight Processing (FP) module, FlyStraight, handles platform movement along a straight flight path if the angle between the platform's current velocity vector and the destination vector is less than 8 degrees. For this case, the assumption is made that only slight modifications to the current flight path are required to maintain a course in the desired direction. Inputs and outputs for FlyStraight are listed in Table 2.2-1, and the steps performed by the module are shown in Figure 2.2-2.

TABLE 2.2-1. Inputs and Outputs for FlyStraight.

Inputs:	
Pos	Aircraft Position Vector
Vel	Aircraft Velocity Vector
Flight	Aircraft Flight Structure Pointer
Aircraft	Aircraft Element Pointer
Dt	Time Step
CmdSpeed	Commanded Speed
VelDir	Desired direction to fly in

TABLE 2.2-1. Inputs and Outputs for FlyStraight. (Contd.)

Outputs:	
Pos	Aircraft Position Vector
Vel	Aircraft Velocity Vector

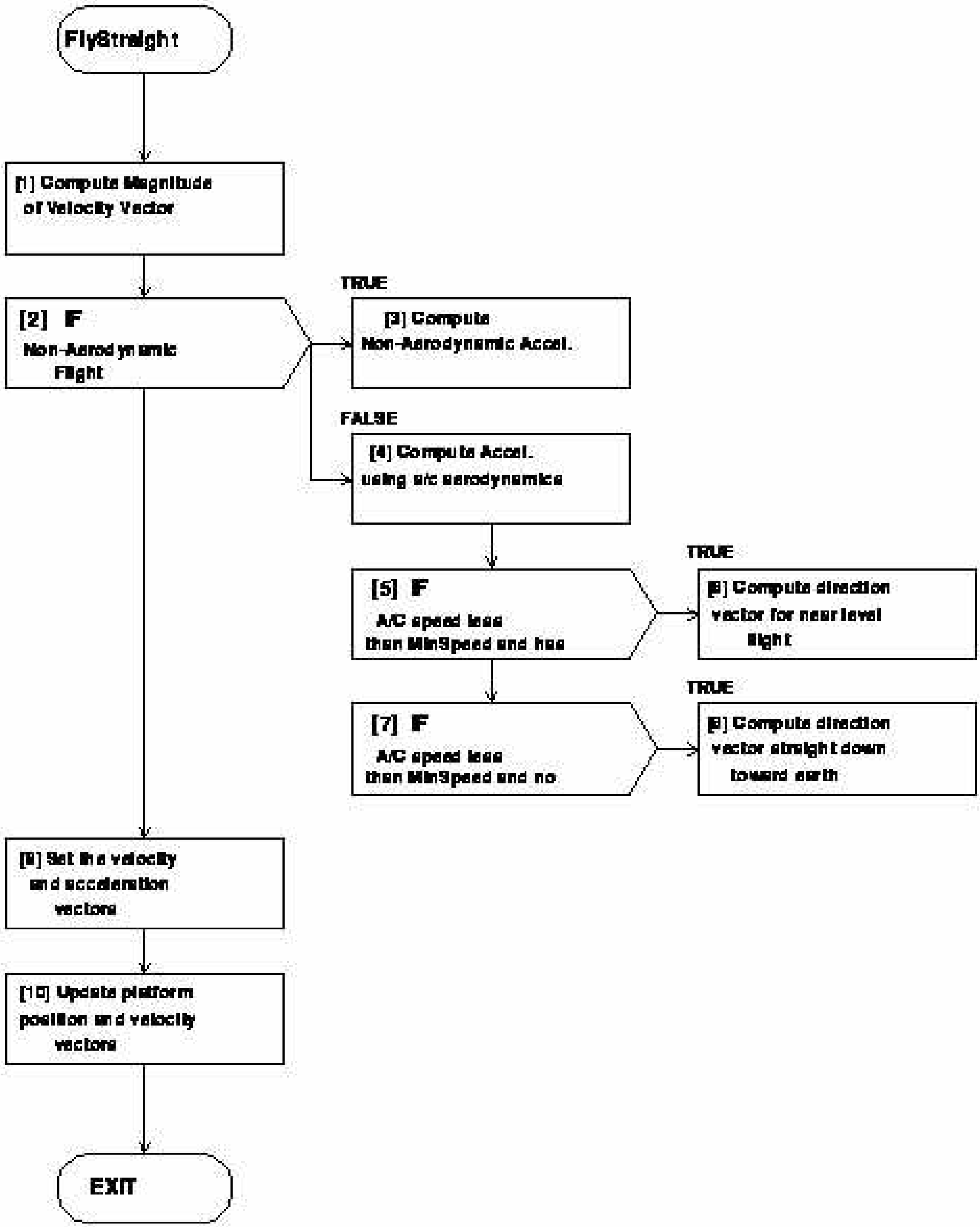

FIGURE 2.2-2. Flow Diagram for FlyStraight.

Block 1. Compute the magnitude of the a/c velocity vector.

Block 2. If the user has specified non-aerodynamic flight, acceleration is computed as the difference between actual and commanded speed over time. For aerodynamic flight, user-defined aircraft aerodynamics parameters are used to compute acceleration.

Block 3. Compute acceleration for non-aerodynamic flight.

Block 4. Compute acceleration for aerodynamic flight.

Block 5. If a/c actual speed is less than minimum speed to maintain flight and a/c has thrust, fly near level flight.

Block 6. Compute direction vector for near level flight.

Block 7. If a/c actual speed is less than minimum speed to maintain flight and a/c has no thrust, fly straight down toward earth.

Block 8. Compute direction vector straight down toward earth.

Block 9. Compute velocity and acceleration vectors.

Block 10. Update a/c position and velocity to reflect new a/c state and end of 0.5 second sub-interval, using current position, re-oriented velocity and acceleration along the new direction vector.

FlyCurve:

The Flight Processing (FP) module, FlyCurve, handles platform movement along a curved flight path if the angle between the platform's current velocity vector and the destination vector is greater than or equal to 8 degrees. Inputs and outputs for FlyCurve are listed in Table 2.2-2, and the steps performed by the module are shown in Figure 2.2-3.

TABLE 2.2-2. Inputs and Outputs for FlyCurve.

Inputs:	
Pos	Aircraft Position Vector
Vel	Aircraft Velocity Vector
Flight	Aircraft Flight Structure Pointer
Aircraft	Aircraft Element Pointer
Dt	Time Step
CmdSpeed	Commanded Speed
TurnDir	Desired direction to fly in
CmdTurnG	Number of G in this turn
CmdRadius	Desired radius for this turn
Outputs:	
Pos	Aircraft Position Vector
Vel	Aircraft Velocity Vector
Roll	Roll angle (degrees)
TurnDir	Direction flown

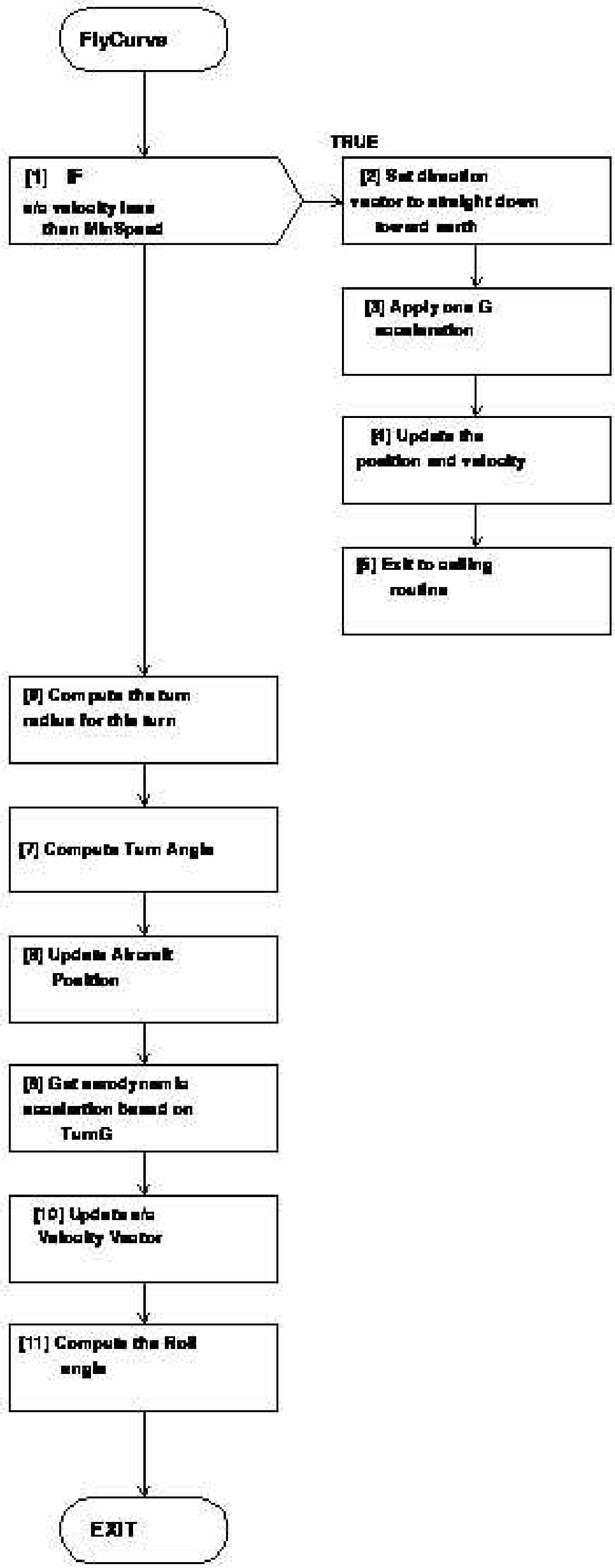

FIGURE 2.2-3. Flow Diagram for FlyCurve.

Block 1. If a/c actual speed is less than minimum speed to maintain flight, fly straight down toward earth

Block 2. Compute direction vector straight down toward earth.

Block 3. Apply one G acceleration along direction vector.

Block 4. Update a/c position and velocity to reflect new a/c state and end of 0.5 second sub-interval

Block 5. Set Roll angle to 0, Exit to the module that called FlyCurve.

Block 6. Compute the turn radius for this turn.

Block 7. Compute the turn angle for this turn.

Block 8. Update a/c position to reflect new a/c state and end of 0.5 second sub-interval.

Block 9. Compute aerodynamic acceleration based on TurnG.

Block 10. Update a/c velocity to reflect new a/c state and end of 0.5 second sub-interval.

Block 11. Compute output roll angle.

FlyVector:

The Flight Processing (FP) module, FlyVector, handles platform movement for the Vector maneuver. Inputs and outputs for FlyVector are listed in Table 2.2-3, and the steps performed by the module are shown in Figure 2.2-4.

TABLE 2.2-3. Inputs and Outputs for FlyVector.

Inputs:	
Scenario	Pointer to current scenario
OpFac	Pointer to the platform to be moved
Outputs:	
Return Status	Returns 0 (zero) for success

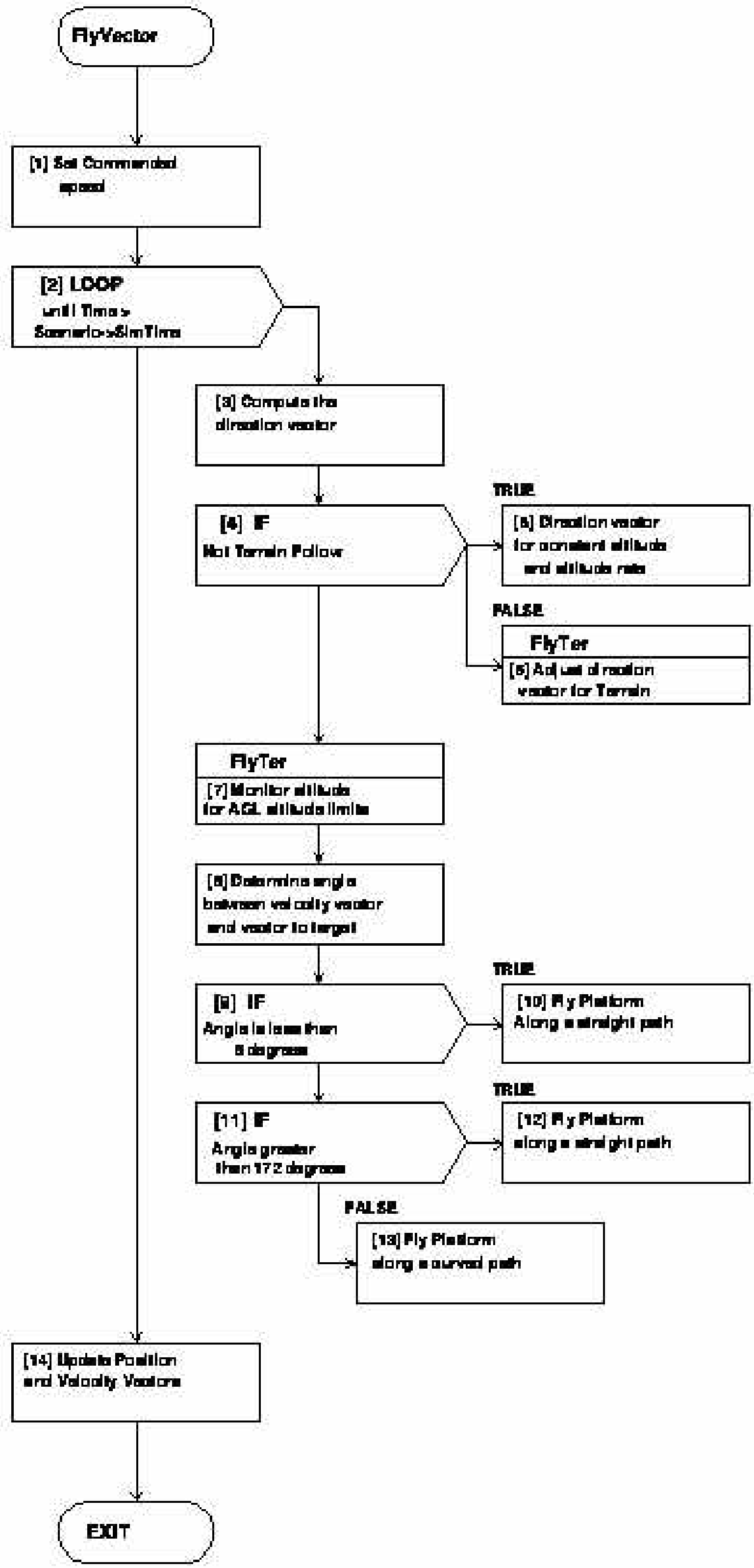

FIGURE 2.2-4. Flow Diagram for FlyVector.

Block 1. Set commanded speed for vector maneuver.

Block 2. The platform movement is propagated in a loop over the scenario interval using smaller 0.5 second sub-intervals.

Block 3. A direction vector is calculated for the vector maneuver to fly platform to the intercept point.

Block 4. A check is made to determine if terrain following is desired for maneuver

Block 5. If terrain following is not desired, the direction vector is calculated that will achieve a constant altitude and altitude rate

Block 6. If terrain following is desired, the FlyTer module is called to adjust the direction vector to fly to desired MSL altitude.

Block 7. The FlyTer module is called to monitor platform altitude for ceiling and floor altitude limits.

Block 8. The angle between the platform velocity vector and the vector to intercept point is calculated so that fly straight or fly curved path algorithms can be chosen.

Block 9. If the angle between the platform velocity vector and the vector to intercept point is less than 8 degrees, the platform will be flown straight.

Block 10. Platform is moved in straight flight path for 0.5 second sub-interval.

Block 11. If the angle between the platform velocity vector and the vector to intercept point is greater than 172 degrees, the platform will be flown straight.

Block 12. The direction vector is adjusted 180 degrees and platform is moved in straight flight path for 0.5 second sub-interval.

Block 13. Platform is moved in curved flight path for 0.5 second sub-interval.

Block 14. Upon completion of loop over 0.5 second sub-intervals, the platforms position and velocity vectors are updated to reflect platform state at the end of this scenario interval.

FlyDrag:

The Flight Processing (FP) module, FlyDrag, handles platform movement for the Drag/Beam maneuver. Inputs and outputs for FlyDrag are listed in Table 2.2-4, and the steps performed by the module are shown in Figure 2.2-5.

TABLE 2.2-4. Inputs and Outputs for FlyDrag.

Inputs:	
Scenario	Pointer to current scenario
OpFac	Pointer to the platform to be moved
Outputs:	
Return Status	Returns 0 (zero) for success

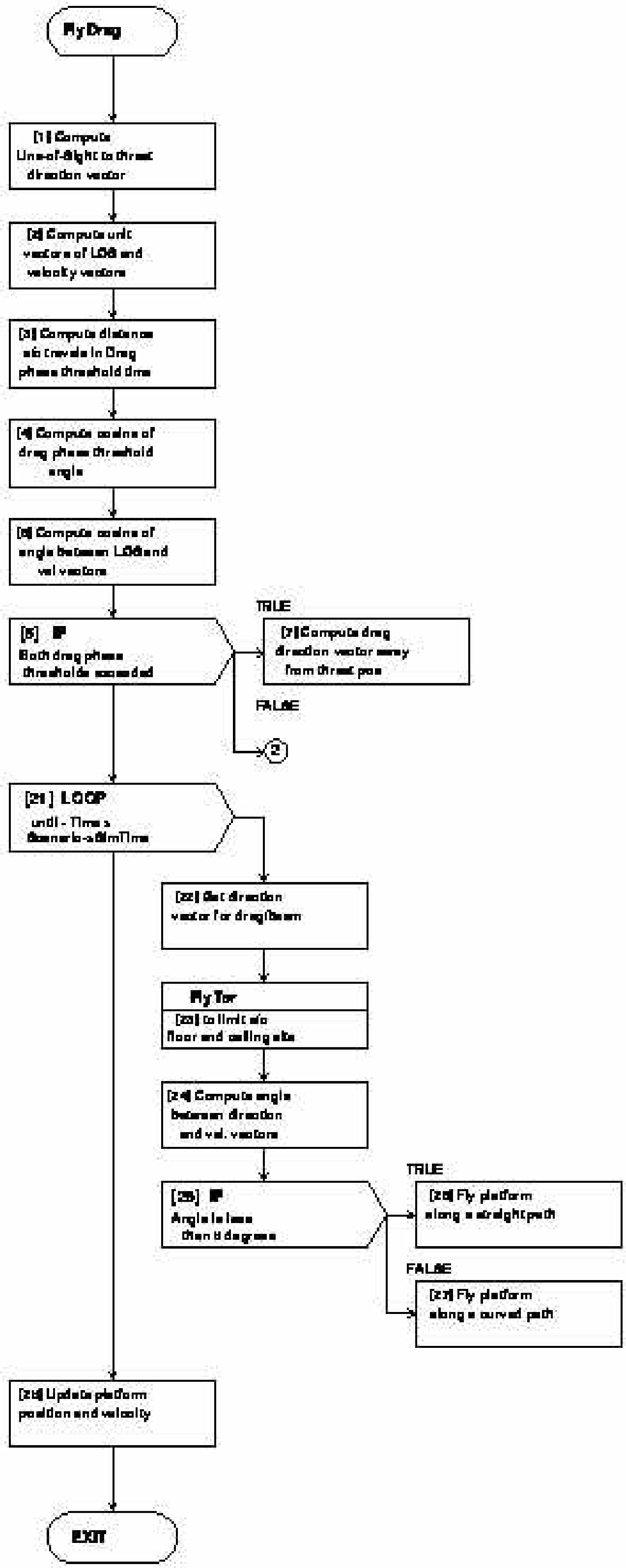

FIGURE 2.2-5. Flow Diagram for FlyDrag.

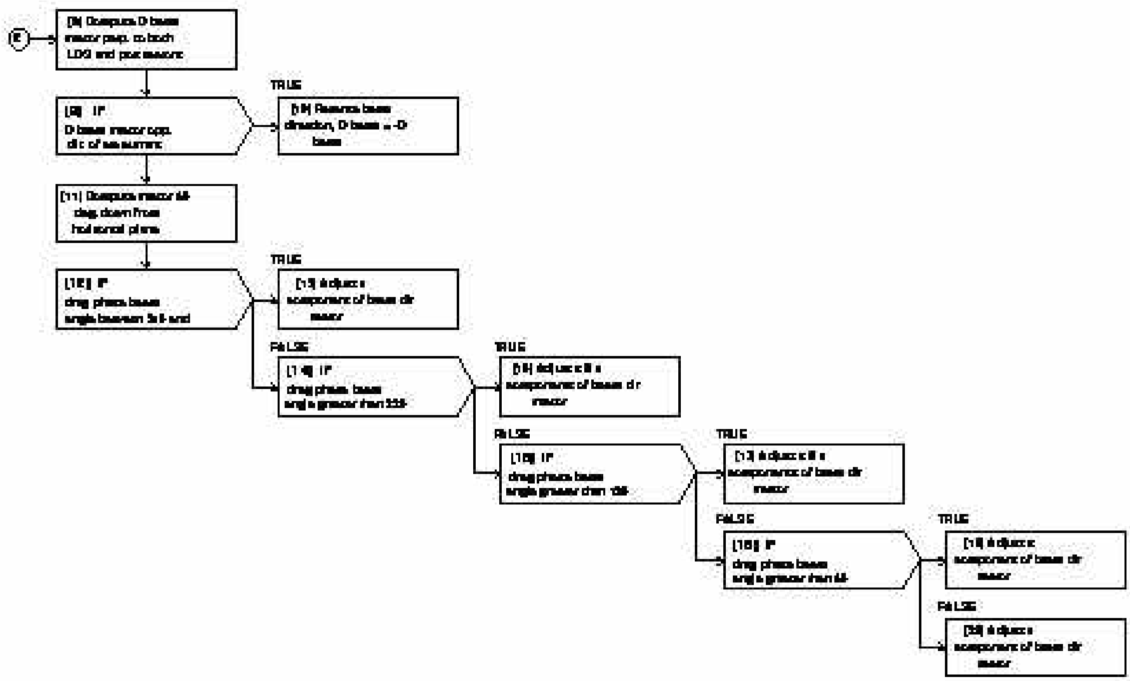

FIGURE 2.2-5. Flow Diagram for FlyDrag. (Contd.)

Block 1. Compute a Line-of-Sight vector from platform to threat.

Block 2. Compute unit vectors for LOS and platform velocity.

Block 3. Compute the range that the platform can fly in the drag phase threshold time.

Block 4. Compute the cosine of the drag phase threshold angle.

Block 5. Using the dot product of unit vectors, compute the cosine of the angle between LOS unit vector and platform unit velocity vector.

Block 6. Check conditions for choosing drag/beam maneuver. If the cosine of the angle between the platform velocity vector and LOS vector is less than the cosine of the drag phase threshold angle and magnitude of the LOS vector is greater than the range the platform can fly during the time threshold, then, both conditions are met and the platform performs a drag maneuver, otherwise, beam.

Block 7. Compute drag maneuver direction vector D drag, as a vector pointing away from the threat.

Block 8. Compare the beam maneuver D beam, as the cross product of the LOS vector and platform position vector.

Block 9. Check D beam against platform current direction.

Block 10. If D beam and platform velocity are opposing, recompute D beam in opposite direction.

Block 11. Compute a vector, D45, 45 degrees down from the local horizontal plane by subtracting the unit attacker position vector from the unit D beam vector. This vector will then be adjusted by user specified beam angle. The user-specified beam angle is an angle relative to 45 degrees down from local horizontal. This angle is used to adjust the x and z components of the desired beam direction vector. The vector components are adjusted by normalizing the user-specified beam angle to the 45 degree reference point. Blocks 12 through 20 are known to be in error. For details, see Section 2.2.5 Known Problems or Anomalies.

Block 12. Check beam angle for 315-360 degree range.

Block 13. Adjust z component of D45. This produces a normalized z component with a change in sign.

Block 14. Check beam angle for >225 degrees.

Block 15. Adjust the x component. This produces a normalized x component with correct sign. Change sign on z component of D45.

Block 16. Check beam angle for >135 degrees.

Block 17. Change sign on the x component. Adjust z component of D45. This produces a normalized z component with correct sign.

Block 18. Check beam angle for >45 degrees.

Block 19. Adjust x component of D45.

Block 20. Adjust z component of D45.

Block 21. The platform movement is propagated in a loop over the scenario interval using smaller 0.5 second sub-intervals.

Block 22. Get direction vector for drag/beam maneuver.

Block 23. Call FlyTer to monitor altitude ceiling/floor limits.

Block 24. Compute angle between the platform velocity vector and the drag/beam direction vector.

Block 25. If the angle between the platform velocity vector and the drag/beam direction vector is less than 8 degrees, the platform will be flown straight.

Block 26. Platform is moved in straight flight path for 0.5 second sub-interval.

Block 27. Platform is moved in curved flight path for 0.5 second sub-interval.

Block 28. Upon completion of loop over 0.5 second sub-intervals, the platform's position and velocity vectors are updated to reflect platform state at the end of this scenario interval.

FlyFPole:

The Flight Processing (FP) module, FlyFPole, handles platform movement for the FPole maneuver. Inputs and outputs for FlyFPole are listed in Table 2.2-5, and the steps performed by the module are shown in Figure 2.2-6.

TABLE 2.2-5. Inputs and Outputs for FlyFPole.

Inputs:	
Scenario	Pointer to current scenario
OpFac	Pointer to the platform to be moved
Outputs:	
Return Status	Returns 0 (zero) for success

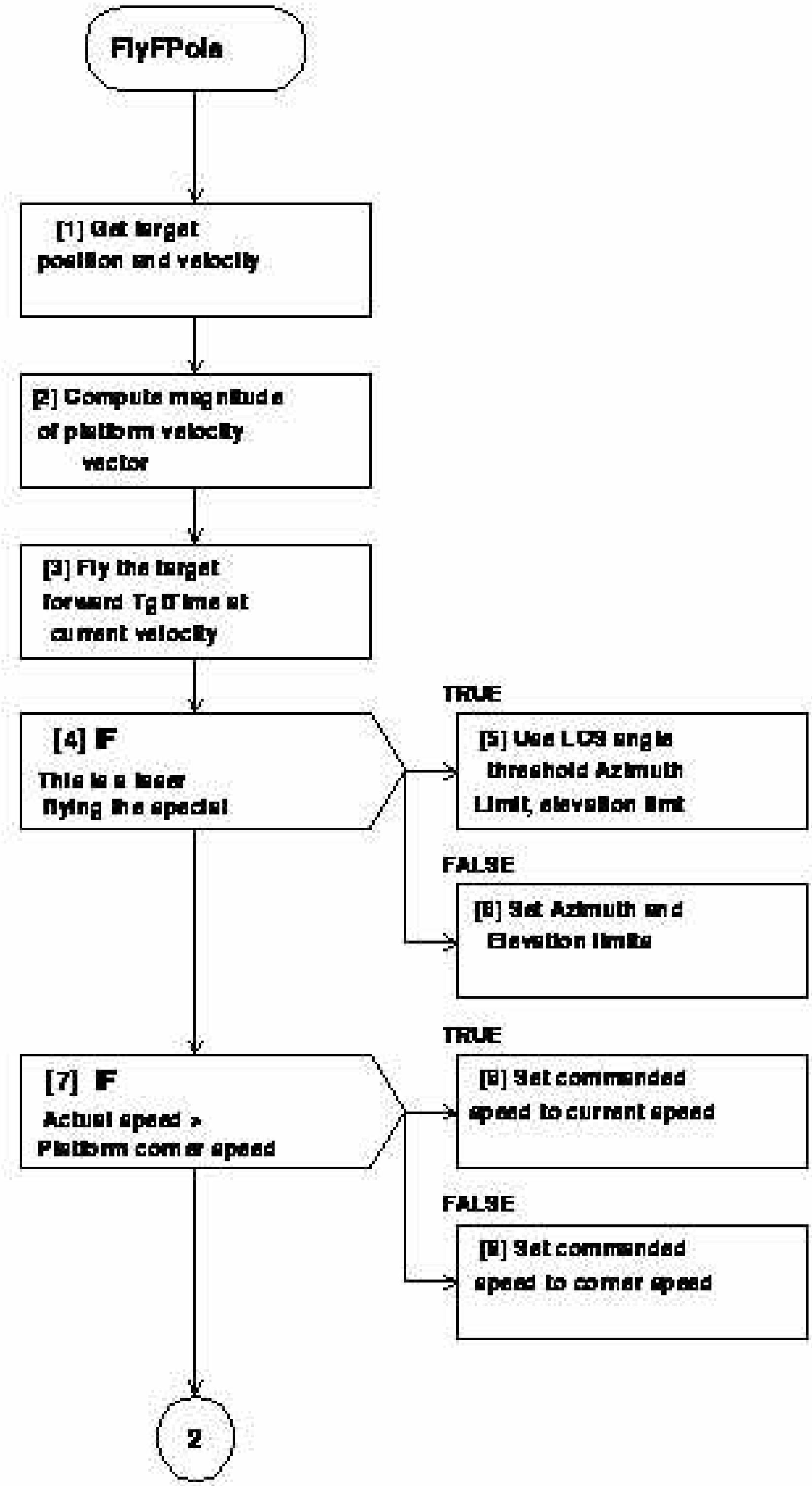

FIGURE 2.2-6. Flow Diagram for FlyFPlole.

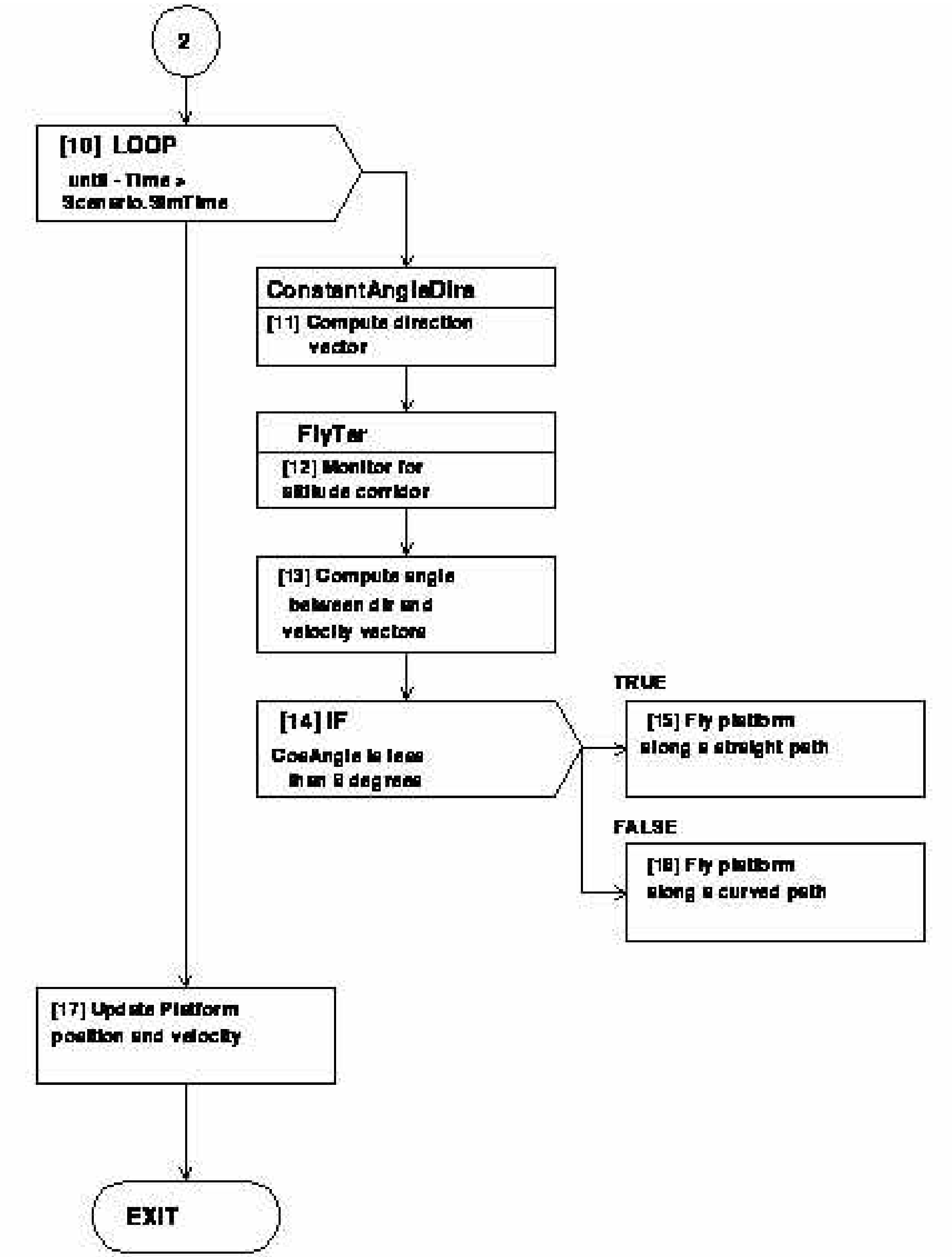

FIGURE 2.2-6. Flow Diagram for FlyFPole. (Contd.)

<u>Block 1.</u> Get the target position and velocity vectors.

<u>Block 2.</u> Compute the speed of the FPole platform.

Block 3. Fly the target forward in time at its current velocity to reflect its position at the end of this scenario interval.

Block 4. A check is made to determine if this is a LASER ruleset platform utilizing angle and altitude maintenance.

Block 5. If this is a LASER platform flying the special maneuver, use Line of Sight angle threshold value for azimuth limit and set elevation limit to 90 degrees.

Block 6. Else set azimuth and elevation limits from values specified in the ruleset engagement parameters.

Block 7. Check to see which speed should be used for maneuver.

Block 8. Set the commanded speed for the maneuver to the current platform speed.

Block 9. Set the commanded speed for the maneuver to the platform corner speed.

Block 10. The platform movement is propagated in a loop over the scenario interval using smaller 0.5 second sub-intervals.

Block 11. ConstantAngleDirection is called to compute the desired FPole maneuver direction vector.

Block 12. FlyTer is called to check altitude floor and ceiling limits.

Block 13. Compute angle between new direction vector and platform velocity vector, so that we can decide whether to use straight flight or curved flight in adopting the new direction vector.

Block 14. If the angle between the platform velocity vector and the vector to intercept point is less than 8 degrees, the platform will be flown straight.

Block 15. Platform is moved in straight flight path for 0.5 second sub-interval.

Block 16. Platform is moved in curved flight path for 0.5 second sub-interval.

Block 17. Upon completion of loop over 0.5 second sub-intervals, the platform's position and velocity vectors are updated to reflect platform state at the end of this scenario interval.

ConstantAngleDirection:

The Flight Processing (FP) module, ConstantAngleDirection, computes the direction vector for the FPole maneuver. Inputs and outputs for ConstantAngleDirection are listed in Table 2.2-6, and the steps performed by the module are shown in Figure 2.1.2-7.

TABLE 2.2-6. Inputs and Outputs for ConstantAngleDirection.

Inputs:	
OpFac	Pointer to platform being moved
AttPos	Attacker Position Vector
TargetPos	Target Position Vector
TVel	Target Velocity Vector
CmdSpeed	Commanded Speed
AzLimit	Azimuth Angle Limit
ElLimit	Elevation Angle Limit
Outputs:	
Dir	Direction Vector for FPole Maneuver

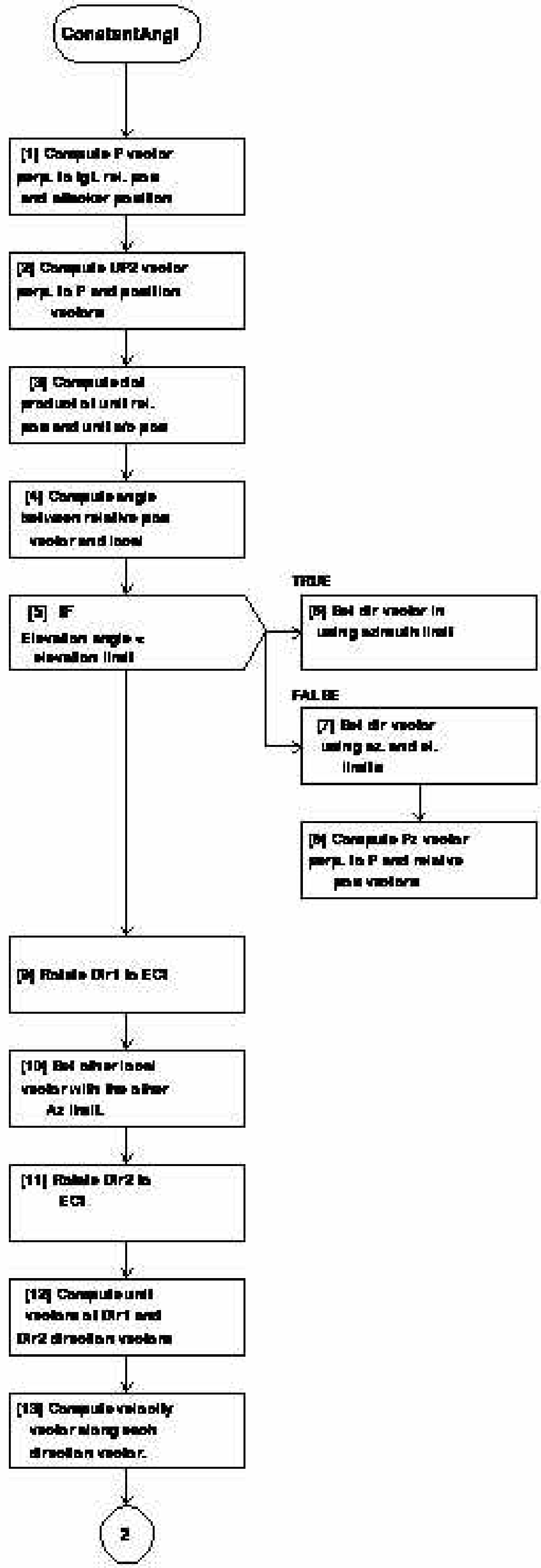

FIGURE 2.2-7. Flow Diagram for ConstantAngleDirection.

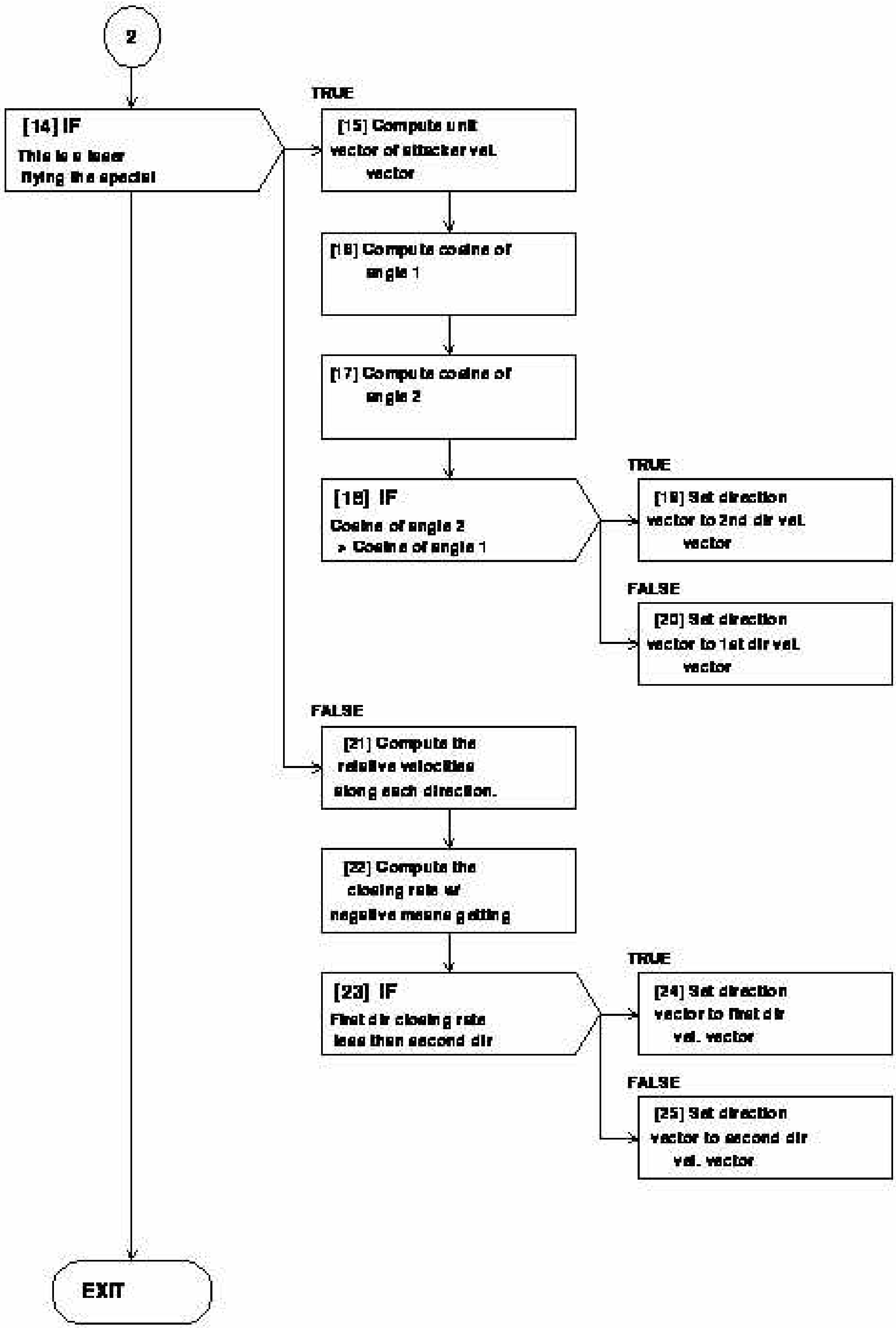

FIGURE 2.2-7. Flow Diagram for ConstantAngleDirection. (Contd.)

Block 1. Compute P vector as the cross product of the attacker position vector and the target position vector relative to the attacker position.

Block 2. Compute vector coplanar to the target relative position vector and perpendicular to the new x-axis vector (i.e. P Vector).

Block 3. Compute the target z component relative to this local coordinate frame by taking the dot product of the unit target relative position vector and the unit attacker position vector.

Block 4. Compute target elevation angle.

Block 5. Check Elevation angle against Elevation Limit.

Block 6. Set direction vector to point in the direction corresponding to the azimuth limit from the target relative position vector and establish transformation matrix for local to ECI coordinate system.

Block 7. Set direction vector using azimuth and elevation limits.

Block 8. Compute Pz vector to represent z-axis and establish local to ECI transformation matrix.

Block 9. RotateDir1 direction vector to ECI coordinate frame.

Block 10. Set other local direction vector (Dir2).

Block 11. RotateDir2 direction vector to ECI coordinate frame.

Block 12. Compute unit vectors for Dir1 and Dir2.

Block 13. Compute velocity vectors, Vel1, Vel2, along each direction vector using the input commanded speed (i.e. CmdSpeed).

Block 14. A check is made to determine if this is a LASER ruleset platform utilizing angle and altitude maintenance.

Block 15. Compute a unit vector of the attackers velocity vector.

Block 16. Use dot product to compute the cosine of the angle between Dir1 direction vector and attackers velocity vector.

Block 17. Use dot product to compute the cosine of the angle between Dir2 direction vector and attackers velocity vector.

Block 18. Compare angles.

Block 19. Set output direction vector for FPole maneuver to Vel2 vector.

Block 20. Set output direction vector for FPole maneuver to Vel1 vector.

Block 21. Compute attacker velocities relative the target velocity along each direction.

Block 22. Compute the closing rate where negative means getting closer.

Block 23. Compute closing rate in each direction.

Block 24. Set output direction vector for FPole maneuver to Vel1 vector.

Block 25. Set output direction vector for FPole maneuver to Vel2 vector.

FlyAvoid:

The Flight Processing (FP) module, FlyAvoid, handles platform movement for the Avoid maneuver. Inputs and outputs for ConstantAngleDirection are listed in Table 2.2-7, and the steps performed by the module are shown in Figure 2.2-8.

TABLE 2.2-7. Inputs and Outputs for FlyAvoid.

Inputs:	
Scenario	Pointer to current scenario
OpFac	Pointer to the platform to be moved
Outputs:	
Return Status	Returns 0 (zero) for success

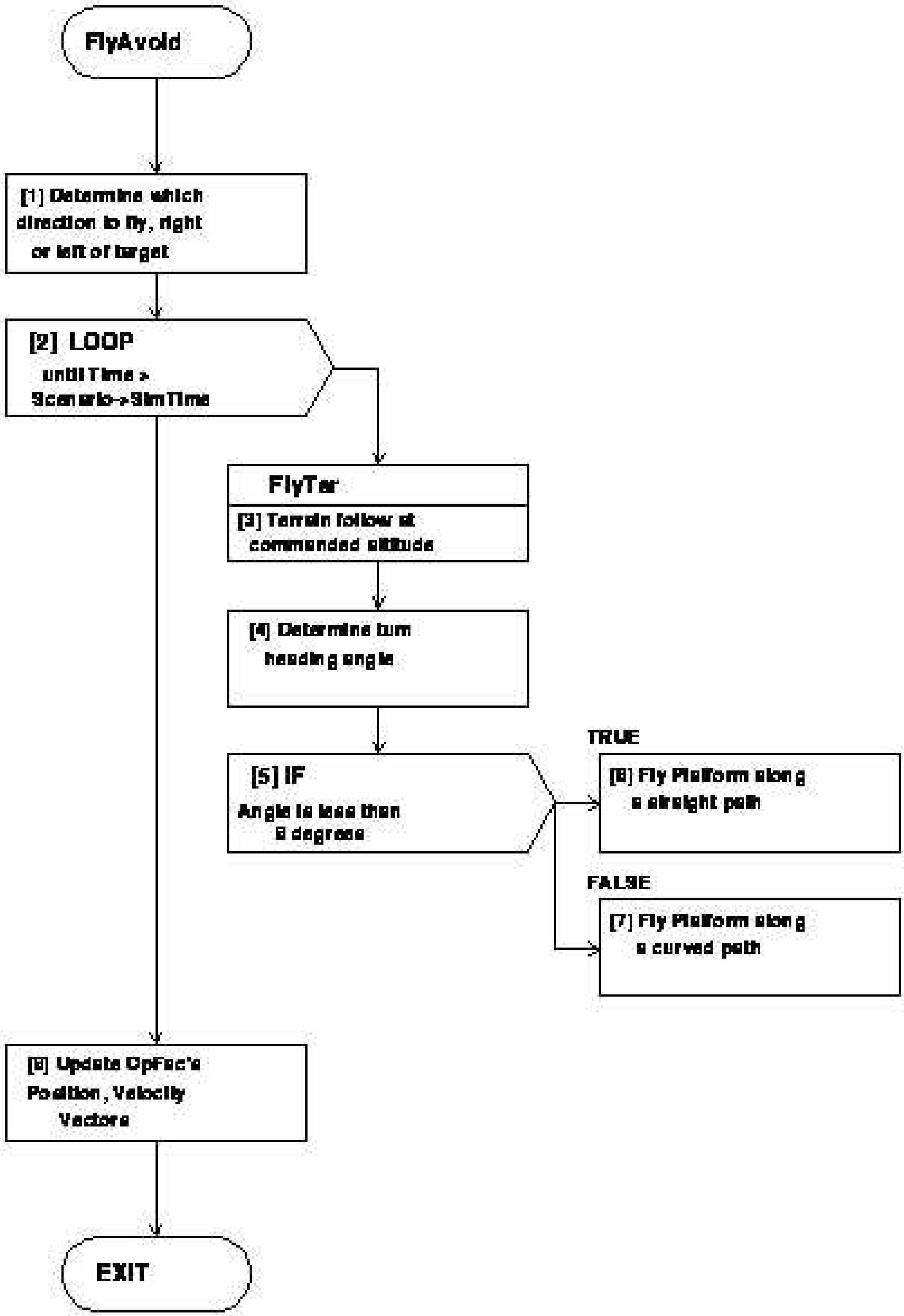

FIGURE 2.2-8. Flow Diagram for FlyAvoid.

Block 1. A determination is made to avoid the target platform to the right or left.

Block 2. The platform movement is propagated in a loop over the scenario interval using smaller 0.5 second sub-intervals.

Block 3. The FlyTer module is called to determine flight path vector for terrain following.

Block 4. The turn angle is determined by taking the dot product of the platform velocity vector and avoid direction vector.

Block 5. A check is made to determine if the platform should adopt the new avoid direction vector by flying straight or by flying in a curved manner until the new vector is achieved. If the turn angle is less than 8 degrees, straight flight is chosen, otherwise, curved flight is chosen.

Block 6. The Platform is moved in a straight path along the avoid direction vector.

Block 7. The Platform is moved in a curved path based on command radius and turn g (i.e. the number of Gs being pulled by the platform in the turn).

Block 8. Upon completion of loop over 0.5 second sub-intervals, the platforms position and velocity vectors are updated to reflect platform state at the end of this scenario interval.

2.2.4 Assumptions and Limitations

Functional Element	Assumptions	Conditions of Applicability
Aircraft	• Lateral forces on aircraft during straight line flight are not modeled. • Lateral forces are balanced during coordinated steady turns • Ordnance weight is included as part of the vehicle's weight and is never decremented. • Aircraft signatures are decremented at weapon launch • Aircraft flight propagation includes no angle of attack sensitivities • Aircraft propagation uses a 3 DOF flight model and assumes a point mass propagated through space • Turns are performed at corner speed assuming max Gs if the A/C is not in CAP, or 3 Gs if it is in CAP	• Computations used in the Flight Processing process to integrate aircraft state as a function of time
Flight Modes	• Drag and Engage modes use target truth position rather than track measurements in setting the direction of flight • AR Tanker flight speed does not account for receiver speed limitations	

Functional Element	Limitations	Conditions of Applicability
Vehicle Movement	• Aircraft flying in formation with tankers during Air Refueling operations have limited defensive reactions. • An interceptor missile cannot be diverted to an alternative target after it has been launched.	

2.2.5 Known Problems or Anomalies

Functional Element	Error	Effects
Vehicle Movement	The calculation of the beam maneuver dive angle is an approximation that assumes an angle can be halved by halving the sine of the angle.	No significant effects result from this error. The computation is only used to determine the signs of the beam vector components.